

فهرست مطالب

بخش ۱: آشنایی با ویژوال C# و مایکروسافت ویژوال استودیو ۲۰۱۳

فصل ۱: به C# خوش آمدید.....	۱۹
شروع به برنامه‌نویسی با محیط ویژوال استودیو ۲۰۱۳.....	۲۰
ایجاد یک برنامه کنسول در ویژوال استودیو ۲۰۱۳.....	۲۰
نوشتن اولین برنامه.....	۲۵
نوشتن کد با استفاده از ابزار IntelliSense مایکروسافت.....	۲۵
ساختن و اجرای برنامه کنسول.....	۲۸
استفاده از فضا‌های نام.....	۳۱
استفاده از نام‌های طولانی.....	۳۲
ایجاد یک برنامه گرافیکی.....	۳۴
ایجاد یک اپلیکیشن گرافیکی در ویژوال استودیو.....	۳۵
ایجاد اینترفیس کاربری.....	۳۸
بررسی اپلیکیشن ویندوز استور.....	۴۵
بررسی برنامه WPF.....	۵۰
اضافه کردن کد به یک برنامه گرافیکی.....	۵۲
نوشتن کد برای دکمه OK.....	۵۲
خلاصه فصل ۱.....	۵۵
فصل ۲: کار کردن با متغیرها، عملگرها و عبارت‌ها.....	۵۷
آشنایی با دستورات.....	۵۸
استفاده از شناسه‌ها.....	۵۸
کلمه‌های کلیدی.....	۵۹
استفاده از متغیرها.....	۶۰
نامگذاری متغیرها.....	۶۰
معرفی متغیرها.....	۶۱
کار کردن با انواع داده اصلی.....	۶۲
متغیرهای محلی بدون مقدار.....	۶۲
نمایش مقادیر انواع داده اصلی.....	۶۳
نمایش مقادیر انواع داده اصلی.....	۶۳
استفاده از انواع داده اصلی در کد.....	۶۵
استفاده از عملگرهای ریاضی.....	۷۰

۷۰	عملگرها و انواع داده
۷۲	بررسی عملگرهای ریاضی
۷۲	اجرای پروژه MathsOperations
۷۳	انجام محاسبات در برنامه MathsOperators
۷۷	آزمایش برنامه MathsOperators
۷۷	کنترل سطح تقدم عملگرها
۷۸	استفاده از وابستگی برای ارزیابی عبارت‌ها
۷۸	وابستگی و عملگر نسبت‌دهی
۷۹	افزایش و کاهش مقادیر متغیرها
۸۰	مکان استفاده از عملگرهای افزایشی و کاهشی
۸۰	معرفی متغیرهای محلی به صورت ضمنی
۸۱	خلاصه فصل ۲
۸۳	فصل ۳: نوشتن متدها و مفهوم قلمروها
۸۴	ایجاد متدها
۸۴	معرفی یک متد
۸۵	برگرداندن داده‌ها از یک متد
۸۶	بررسی تعریف‌های متد
۸۷	فراخواندن متدها
۸۷	روش فراخواندن یک متد
۸۹	بررسی فراخوان‌های متد
۹۰	آشنایی با قلمروها
۹۰	تعریف قلمرو محلی
۹۱	تعریف قلمرو سطح کلاس
۹۲	اورلود کردن متدها
۹۲	نوشتن متدها
۹۳	ایجاد منطق برای برنامه
۹۳	نوشتن متدها با استفاده از ویزارد Generate Method Stub
۹۷	آزمایش برنامه
۹۷	استفاده از دیباگر ویژوال استودیو
۱۰۰	استفاده از پارامترهای دلخواه و آرگومان‌های دارای نام
۱۰۲	تعریف پارامترهای دلخواه
۱۰۲	دادن آرگومان‌های دارای نام
۱۰۳	حل ابهام بین پارامترهای دلخواه و آرگومان‌های دارای نام
۱۰۴	تعریف و فراخواندن یک متد با پارامترهای دلخواه
۱۰۷	خلاصه فصل ۳
۱۰۹	فصل ۴: استفاده از دستورات تصمیم‌گیری
۱۱۰	معرفی متغیرهای بولین
۱۱۰	استفاده از عملگرهای بولین

۱۱۰	آشنایی با عملگرهای تساوی و نسبی
۱۱۱	آشنایی با عملگرهای منطق شرطی
۱۱۲	کوتاه کردن مدار
۱۱۲	خلاصه ای از قدم‌ها و وابستگی‌های عملگرها
۱۱۳	استفاده از دستورات if برای تصمیم‌گیری
۱۱۳	چگونگی استفاده از دستور if
۱۱۵	استفاده از بلاک‌ها برای گروه‌بندی دستورات
۱۱۶	آبشاری کردن دستورات if
۱۱۷	نوشتن دستورات if
۱۲۳	استفاده از دستورات Switch
۱۲۳	چگونگی استفاده از دستور switch
۱۲۵	قوانین دستور switch
۱۲۶	نوشتن دستورات switch
۱۲۸	خلاصه فصل ۴
۱۳۱	فصل ۵: استفاده از نسبت‌دهی ترکیبی و دستورات تکرار شدنی
۱۳۲	استفاده از عملگرهای نسبت‌دهی ترکیبی
۱۳۳	دستورات while
۱۳۵	نوشتن یک دستور while
۱۳۹	نوشتن دستورات for
۱۴۰	قلمرو دستور
۱۴۱	استفاده از دستور do
۱۴۳	نوشتن یک دستور do
۱۴۵	انجام دستور do به صورت مرحله به مرحله
۱۵۰	خلاصه فصل ۵
۱۵۱	فصل ۶: مدیریت خطاها و استثناها
۱۵۲	کار کردن با خطاها
۱۵۲	استفاده از دستورات Try و Catch
۱۵۴	استثناهای اداره نشده
۱۵۵	استفاده از چند اداره کننده catch
۱۵۵	دریافت چند استثنا
۱۵۷	دیدن چگونگی گزارش ویندوز برای استثناهای اداره نشده
۱۵۹	نوشتن یک بلاک دستور try/catch
۱۶۲	جمع کردن استثناها
۱۶۲	جمع کردن استثناها
۱۶۴	استفاده از ریاضیات عدد صحیح چک شده و چک نشده
۱۶۵	نوشتن دستورات چک شده
۱۶۵	نوشتن عبارت‌های چک شده
۱۶۶	استفاده از عبارت‌های چک شده
۱۶۹	انداختن استثناها

۱۷۰.....	انداختن یک استثنا.....
۱۷۲.....	دریافت یک استثنا.....
۱۷۳.....	دریافت استثناهای اداره نشده.....
۱۷۴.....	استفاده از یک بلاک finally.....
۱۷۵.....	خلاصه فصل ۶.....

بخش ۲: آشنایی با مدل شی گرای C#

۱۷۹.....	فصل ۷: ایجاد و مدیریت کلاس ها و اشیا
۱۸۰.....	آشنایی با کلاس بندی.....
۱۸۰.....	دلیل جداسازی.....
۱۸۱.....	تعریف و استفاده از یک کلاس.....
۱۸۲.....	کنترل دسترسی.....
۱۸۴.....	کار کردن با سازنده ها.....
۱۸۵.....	اورلود کردن سازنده ها.....
۱۸۷.....	نوشتن سازنده ها و ایجاد اشیا.....
۱۹۲.....	نوشتن و فراخواندن متدهای نمونه.....
۱۹۴.....	آشنایی با متدها و داده های استاتیک.....
۱۹۵.....	ایجاد یک فیلد اشتراکی.....
۱۹۶.....	ایجاد یک فیلد استاتیک ثابت با استفاده از کلمه کلیدی const.....
۱۹۷.....	آشنایی با کلاس های استاتیک.....
۱۹۷.....	نوشتن اعضای static و فراخواندن متدهای static.....
۲۰۰.....	کلاس های ناشناس.....
۲۰۱.....	خلاصه فصل ۷.....
۲۰۳.....	فصل ۸: آشنایی با مقادیر و ارجاع ها
۲۰۴.....	کپی کردن متغیرها و کلاس ها.....
۲۰۷.....	استفاده از پارامترهای مقداری و ارجاعی.....
۲۱۰.....	آشنایی با مقادیر خالی و انواع قابل خالی بودن.....
۲۱۱.....	استفاده از انواع قابل خالی بودن.....
۲۱۲.....	آشنایی با مشخصه های انواع قابل خالی بودن.....
۲۱۳.....	استفاده از پارامترهای ref و out.....
۲۱۴.....	ایجاد پارامترهای ref.....
۲۱۵.....	ایجاد پارامترهای out.....
۲۱۶.....	استفاده از پارامترهای ref.....
۲۱۷.....	مدیریت حافظه در برنامه ها.....
۲۱۷.....	پشته.....
۲۱۸.....	هیپ.....
۲۱۹.....	کلاس System.Object.....

۲۲۰	 بسته‌بندی
۲۲۱	 از بسته‌بندی خارج کردن
۲۲۳	 تبدیل امن داده‌ها
۲۲۳	 عملگر is
۲۲۴	 عملگر as
۲۲۴	 خلاصه فصل ۸
۲۲۷	فصل ۹: ایجاد انواع مقداری با ساختارها و انواع شمارشی
۲۲۸	 کار با شمارشی‌ها
۲۲۸	 معرفی یک شمارشی
۲۲۸	 استفاده از یک شمارشی
۲۲۹	 انتخاب مقادیر مستقیم شمارشی
۲۳۰	 انتخاب نوع درونی یک شمارشی
۲۳۰	 ایجاد یک شمارشی
۲۳۲	 کارکردن با ساختارها
۲۳۴	 معرفی یک ساختار
۲۳۵	 تفاوت کلاس و ساختار
۲۳۷	 معرفی متغیرهای ساختاری
۲۳۸	 مقداردهی اولیه ساختارها
۲۴۰	 ایجاد و استفاده از یک نوع ساختاری
۲۴۳	 کپی کردن متغیرهای ساختاری
۲۴۴	 مقایسه رفتار یک ساختار و یک کلاس
۲۴۷	 خلاصه فصل ۹
۲۴۹	فصل ۱۰: استفاده از آرایه‌ها
۲۵۰	 معرفی و ایجاد یک آرایه
۲۵۰	 معرفی متغیرهای آرایه‌ای
۲۵۱	 ایجاد یک نمونه آرایه‌ای
۲۵۲	 دادن مقدار به یک آرایه و استفاده از آن
۲۵۲	 ایجاد یک آرایه با نوع ضمنی
۲۵۳	 دسترسی به یک عنصر منفرد آرایه
۲۵۴	 تکرار روی یک آرایه
۲۵۵	 دادن آرایه به عنوان یک پارامتر به یک متد و دریافت آن به عنوان مقدار برگشتی
۲۵۷	 کپی کردن آرایه‌ها
۲۵۹	 استفاده از آرایه‌های چند بعدی
۲۶۰	 ایجاد آرایه‌های ناهموار
۲۶۱	 استفاده از آرایه‌ها برای پیاده‌سازی یک بازی
۲۷۲	 خلاصه فصل ۱۰
۲۷۳	فصل ۱۱: آشنایی با آرایه‌های پراامتری
۲۷۴	 اورلود کردن
۲۷۵	 استفاده از آرگومان‌های آرایه‌ای

۲۷۶	معرفی یک آرایه params
۲۷۸	استفاده از params object[]
۲۸۰	استفاده از یک آرایه params
۲۸۰	نوشتن یک متد با پارامتر آرایه params
۲۸۲	آزمایش متد Util.Sum
۲۸۳	مقایسه آرایه‌های پارامتری و پارامترهای دلخواه
۲۸۳	مقایسه یک آرایه params و پارامترهای دلخواه
۲۸۵	خلاصه فصل ۱۱
۲۸۷	فصل ۱۲: کار کردن با وراثت
۲۸۸	وراثت چیست؟
۲۸۹	استفاده از وراثت
۲۹۱	کلاس System.Object
۲۹۲	فراخواندن سازنده‌های کلاس مبنا
۲۹۳	نسبت دادن کلاس‌ها
۲۹۵	معرفی متدهای جدید
۲۹۷	معرفی متدهای مجازی
۲۹۸	معرفی متدهای بازنویسی
۳۰۱	آشنایی با دسترسی حفاظت شده
۳۰۲	ایجاد یک سلسله مراتب از کلاس‌ها
۳۰۸	آشنایی با متدهای بسطی
۳۰۹	ایجاد یک متد بسطی
۳۱۲	خلاصه فصل ۱۲
۳۱۵	فصل ۱۳: ایجاد اینترفیس‌ها و تعریف کلاس‌های انتزاعی
۳۱۶	آشنایی با اینترفیس‌ها
۳۱۷	تعریف یک اینترفیس
۳۱۷	پیاده‌کردن یک اینترفیس
۳۱۹	ارجاع به یک کلاس از طریق اینترفیس آن
۳۲۰	کار کردن با چند اینترفیس
۳۲۰	پیاده‌سازی صریح یک اینترفیس
۳۲۲	محدودیت‌های اینترفیس
۳۲۳	تعریف و استفاده از اینترفیس‌ها
۳۲۳	تعریف اینترفیس‌های IDraw و IColor
۳۲۵	ایجاد کلاس‌های Square و Circle و پیاده‌سازی اینترفیس‌ها
۳۳۰	آزمایش کلاس‌های Square و Circle
۳۳۴	کلاس‌های انتزاعی
۳۳۶	متدهای انتزاعی
۳۳۶	کلاس‌های بسته
۳۳۷	متدهای بسته

۳۳۷	 پیاده‌سازی و استفاده از یک کلاس انتزاعی
۳۳۷	 / ایجاد کلاس انتزاعی DrawingShape
۳۴۰	 تغییر کلاس‌های Square و Circle برای ارث بردن از کلاس DrawingShape
۳۴۲	 خلاصه فصل ۱۳
۳۴۵	 فصل ۱۴: استفاده از ابزار زباله‌روب و مدیریت منابع
۳۴۶	 عمر و زمان یک شی
۳۴۷	 نوشتن تخریب‌گر
۳۴۹	 چرا از زباله‌روب استفاده می‌شود؟
۳۵۰	 زباله‌روب چگونه کار می‌کند
۳۵۱	 توصیه‌ها
۳۵۱	 مدیریت منابع
۳۵۱	 متدهای disposal
۳۵۲	 متد disposal اداره کننده استثنا
۳۵۳	 دستور using و اینترفیس IDisposable
۳۵۵	 فراخواندن متد Dispose از یک تخریب‌گر
۳۵۷	 پیاده‌سازی حذف ایمن در برابر استثنا
۳۵۸	 / ایجاد یک کلاس ساده که از یک تخریب‌گر استفاده می‌کند
۳۶۰	 پیاده‌سازی اینترفیس IDisposable
۳۶۱	 جلوگیری از حذف یک شی بیش از یک بار
۳۶۴	 اطمینان از این که یک شی بعد از یک استثنا حذف می‌شود
۳۶۶	 خلاصه فصل ۱۴

بخش ۳: تعریف انواع قابل بسط با C#

۳۷۱	 فصل ۱۵: استفاده از مشخصات
۳۷۲	 اعمال جداسازی با استفاده از متدها
۳۷۴	 مشخصات چه هستند؟
۳۷۷	 استفاده از مشخصه‌ها
۳۷۷	 مشخصات تنها خواندنی
۳۷۸	 مشخصه‌های تنها نوشتنی
۳۷۹	 دسترسی به مشخصه‌ها
۳۷۹	 آشنایی با محدودیت‌های مشخصه
۳۸۱	 معرفی مشخصه‌های اینترفیس
۳۸۳	 عوض کردن متدها با مشخصات
۳۸۳	 استفاده از مشخصات
۳۸۸	 ایجاد مشخصه‌های خودکار
۳۸۹	 مقداردهی اولیه اشیا با استفاده از مشخصه‌ها
۳۹۲	 تعریف مشخصه‌های خودکار و استفاده از مقداردهی اولیه شی
۳۹۴	 خلاصه فصل

فصل ۱۶: شاخص دهنده‌ها ۳۹۷

۳۹۸	شاخص دهنده چیست؟
۳۹۸	مثالی که از شاخص دهنده استفاده نمی‌کند
۴۰۰	همان مثال با استفاده از شاخص دهنده‌ها
۴۰۲	آشنایی با متدهای دسترسی شاخص دهنده‌ها
۴۰۳	مقایسه شاخص دهنده‌ها و آرایه‌ها
۴۰۵	شاخص دهنده‌ها در اینترفیس‌ها
۴۰۷	استفاده از شاخص دهنده‌ها در یک برنامه ویندوز
۴۰۷	آشنا شدن با برنامه
۴۰۹	نوشتن شاخص دهنده‌ها
۴۱۱	فراخواندن شاخص دهنده‌ها
۴۱۲	آزمایش برنامه
۴۱۳	خلاصه فصل ۱۶

فصل ۱۷: آشنایی با جنریک‌ها ۴۱۵

۴۱۶	مشکل نوع Object
۴۲۰	راه حل جنریک
۴۲۳	کلاس‌های جنریک و کلاس‌های عمومی شده
۴۲۴	جنریک‌ها و قیدها
۴۲۴	ایجاد یک کلاس جنریک
۴۲۴	تئوری درخت‌های دوگانه
۴۲۷	ایجاد یک کلاس درخت دوگانه با استفاده از جنریک‌ها
۴۲۹	ایجاد کلاس <code>Tree<TItem></code>
۴۳۴	آزمایش کلاس <code>Tree<TItem></code>
۴۳۷	ایجاد یک متد جنریک
۴۳۸	تعریف یک متد جنریک برای ساختن یک درخت دوگانه
۴۳۸	نوشتن متد <code>InsertIntoTree</code>
۴۳۹	آزمایش متد <code>InsertIntoTree</code>
۴۴۰	اینترفیس‌های جنریک و واریانسی
۴۴۲	اینترفیس‌های کواریانسی
۴۴۴	اینترفیس‌های کانتروارینسی
۴۴۶	خلاصه فصل ۱۷

فصل ۱۸: استفاده از مجموعه‌ها ۴۴۹

۴۵۰	کلاس‌های مجموعه‌ای چه هستند؟
۴۵۱	کلاس <code>List<T></code>
۴۵۴	کلاس مجموعه <code>LinkedList<T></code>
۴۵۶	کلاس مجموعه‌ای <code>Queue<T></code>
۴۵۷	کلاس مجموعه‌ای <code>Stack<T></code>
۴۵۹	کلاس مجموعه‌ای <code>Dictionary<TKey, TValue></code>

۴۶۰		SortedList<TKey,TValue> کلاس مجموعه‌ای
۴۶۲		HashSet<T> کلاس مجموعه‌ای
۴۶۴		استفاده از مقداردهی اولیه مجموعه‌ها
۴۶۴		متدهای Find، گزاره‌ها و عبارت‌های لاند
۴۶۶		مقایسه آرایه‌ها و مجموعه‌ها
۴۶۷		استفاده از کلاس‌های مجموعه‌ای برای یک بازی
۴۶۷		استفاده از مجموعه‌ها برای پیاده کردن یک بازی
۴۷۲		خلاصه فصل ۱۸
۴۷۵		فصل ۱۹: مجموعه‌های قابل شمارش
۴۷۶		شمردن عناصر درون یک مجموعه
۴۷۷		پیاده کردن یک شمارشگر
۴۷۷		ایجاد یک کلاس TreeEnumerator
۴۸۲		پیاده کردن اینترفیس IEnumerable
۴۸۲		پیاده کردن IEnumerale<TItem> در کلاس Tree<TItem>
۴۸۲		آزمایش شمارشگر
۴۸۴		پیاده کردن یک شمارشگر با استفاده از یک تکرارگر
۴۸۴		یک تکرارگر ساده
۴۸۷		تعریف یک شمارشگر برای کلاس Tree<TItem> با استفاده از یک تکرارگر
۴۸۷		اضافه کردن یک شمارشگر به کلاس Tree<TItem>
۴۸۸		آزمایش شمارشگر جدید
۴۸۹		خلاصه فصل ۱۹
۴۹۱		فصل ۲۰: جدا کردن منطق برنامه و اداره رویدادها
۴۹۲		آشنایی با عامل‌ها
۴۹۳		مثال‌هایی از عامل‌ها در کتابخانه کلاس فریم‌ورک دات‌نت
۴۹۵		سناریوی کارخانه خودکار
۴۹۵		پیاده‌سازی سیستم کنترل کارخانه بدون استفاده از عامل‌ها
۴۹۶		پیاده‌سازی کارخانه با استفاده از یک عامل
۴۹۹		معرفی و استفاده از عامل‌ها
۵۰۰		بررسی برنامه Wide World Importers
۵۰۴		ایجاد جز CheckoutController
۵۰۸		آزمایش برنامه
۵۰۸		عبارت‌های لاند و عامل‌ها
۵۰۹		ایجاد یک تطبیق‌دهنده متد
۵۰۹		فرم‌های عبارت‌های لاند
۵۱۱		استفاده از رویدادها
۵۱۱		معرفی یک رویداد
۵۱۲		ثبت نام در یک رویداد
۵۱۳		از ثبت نام خارج کردن از یک رویداد
۵۱۳		فعال کردن یک رویداد

۵۱۴.....	آشنایی با رویدادهای اینترفیس کاربری
۵۱۶.....	استفاده از رویدادها
۵۱۷.....	اضافه کردن یک رویداد به کلاس CheckoutController
۵۲۴.....	خلاصه فصل ۲۰
۵۲۷....	فصل ۲۱: کوئری دادن به داده‌های درون حافظه با استفاده از عبارتهای کوئری
۵۲۸.....	کوئری یکپارچه با زبان چیست؟
۵۲۹.....	استفاده از لینک در یک برنامه C#
۵۳۱.....	انتخاب داده‌ها
۵۳۴.....	فیلتر کردن داده‌ها
۵۳۴.....	مرتب کردن، گروه‌بندی و جمع زدن داده‌ها
۵۳۷.....	الحاق داده‌ها
۵۳۸.....	استفاده از عملگرهای کوئری
۵۴۱.....	دادن کوئری در اشیای Tree<TItem>
۵۴۱.....	دریافت داده‌ها از یک BinaryTree با استفاده از متدهای بسطی
۵۴۶.....	دریافت داده‌ها از یک BinaryTree با عملگرهای کوئری
۵۴۸.....	لینک و ارزیابی معوق
۵۴۸.....	بررسی تاثیر ارزیابی معوق و کش شده یک کوئری لینک
۵۵۲.....	خلاصه فصل ۲۱
۵۵۵.....	فصل ۲۲: اورلود کردن عملگرها
۵۵۶.....	آشنایی با عملگرها
۵۵۶.....	قیده‌های عملگری
۵۵۷.....	عملگرهای اورلود شده
۵۵۸.....	ایجاد عملگرهای متقارن
۵۶۰.....	عملگرها و قابلیت استفاده از آنها در زبان‌های مختلف
۵۶۱.....	آشنایی با ارزیابی نسبت‌دهی ترکیبی
۵۶۱.....	معرفی عملگرهای افزایشی و کاهشی
۵۶۲.....	عملگرهای مقایسه‌ای در ساختارها و کلاس‌ها
۵۶۳.....	تعریف جفت‌های عملگری
۵۶۴.....	پیاده کردن عملگرها
۵۶۵.....	ایجاد کلاس Complex و پیاده‌سازی عملگرهای ریاضی
۵۶۹.....	پیاده کردن عملگرهای تساوی
۵۷۲.....	آشنایی با عملگرهای تبدیل
۵۷۲.....	ایجاد تبدیل‌های داخلی
۵۷۳.....	پیاده کردن عملگرهای تبدیل تعریف شده توسط کاربر
۵۷۴.....	ایجاد عملگرهای متقارن با روش دیگر
۵۷۵.....	نوشتن عملگرهای تبدیل
۵۷۵.....	پیاده کردن عملگرهای تبدیل
۵۷۸.....	خلاصه فصل ۲۲

بخش ۴: ساخت برنامه‌های ویندوز ۸

فصل ۲۳: بهبود عملکرد با استفاده از وظایف ۵۸۱

۵۸۲	 پیاده‌سازی چندوظیفگی با فریم‌ورک دانت
۵۸۲	 وظایف، ریسمان‌ها و ThreadPool
۵۸۴	 ایجاد، اجرا و کنترل وظایف
۵۸۷	 استفاده از کلاس Task برای پیاده کردن موازی بودن
۵۸۷	 بررسی برنامه تک ریسمان GraphDemo
۵۹۴	 تغییر برنامه GraphDemo برای استفاده از اشیای Task
۵۹۷	 استفاده از کلاس Parallel
۵۹۹	 استفاده از کلاس Parallel برای موازی کردن عملیات‌ها در برنامه GraphData
۶۰۲	 زمانی که نباید از کلاس Parallel استفاده کرد
۶۰۵	 کنسل کردن وظایف و اداره استثناها
۶۰۵	 چگونگی کنسل کردن مشارکتی
۶۰۷	 اضافه کردن عملکرد کنسل کردن به برنامه GraphDemo
۶۱۳	 نمایش وضعیت هر وظیفه
۶۱۵	 اطلاع درباره کنسل شده و اداره استثنای OperationCanceledException
۶۱۸	 ادامه دادن به کار با وظایف کنسل شده و خطادار
۶۱۹	 خلاصه فصل ۲۳

فصل ۲۴: بهبود زمان پاسخ با انجام عملیات‌های غیرهمزمان ۶۲۱

۶۲۲	 پیاده کردن متدهای غیرهمزمان
۶۲۳	 تعریف متدهای غیرهمزمان: مسئله
۶۲۶	 تعریف متدهای غیرهمزمان: راه حل
۶۲۹	 تغییر برنامه GraphDemo برای استفاده از یک متد غیرهمزمان
۶۳۲	 تعریف متدهای غیرهمزمان که مقداری را بر می‌گردانند
۶۳۳	 متدهای غیرهمزمان و APIهای Windows Runtime
۶۳۵	 استفاده از پی‌لینک برای موازی سازی دسترسی داده‌ها
۶۳۶	 استفاده از پی‌لینک برای بهبود عملکرد هنگام تکرار روی یک مجموعه
۶۳۶	 موازی کردن یک کوثری لینک روی یک مجموعه ساده
۶۳۸	 موازی کردن یک کوثری لینک که دو مجموعه را به یکدیگر وصل می‌کند
۶۴۰	 کنسل کردن یک کوثری پی‌لینک
۶۴۱	 همزمان کردن دسترسی به داده‌ها
۶۴۵	 قفل کردن داده‌ها
۶۴۵	 اشیای اولیه همزمانی برای وظایف هماهنگ
۶۴۷	 کنسل کردن همزمانی
۶۴۸	 کلاس‌های مجموعه‌ای همزمان
۶۴۹	 استفاده از یک مجموعه همزمان و یک قفل برای ایجاد دسترسی به داده امن برای ریسمان
۶۵۱	 محاسبه عدد π با استفاده از یک ریسمان

محاسبه π با استفاده از وظایف موازی.....	۶۵۵
استفاده از یک مجموعه ایمن برای ریسمان.....	۶۵۷
استفاده از قفل‌ها برای سریالی کردن فراخوان‌های متد.....	۶۵۹
خلاصه سریع فصل ۲۴.....	۶۶۰
فصل ۲۵: پیاده‌سازی اینترفیس کاربری برای یک اپلیکیشن ویندوز استور.....	۶۶۳
یک اپلیکیشن ویندوز استور چیست؟.....	۶۶۴
استفاده از یک الگوی Blank App برای ایجاد یک اپلیکیشن ویندوز استور.....	۶۶۶
ایجاد اپلیکیشن Adventure Workds Customers.....	۶۶۷
پیاده کردن یک اینترفیس کاربری منقطع.....	۶۷۱
چیدمان صفحه برای اپلیکیشن Customers.....	۶۷۱
پیاده‌سازی یک چیدمان جدولی با یک کنترل Grid.....	۶۸۵
تغییر چیدمان برای منقطع کردن اپلیکیشن نسبت به فاکتورهای فرم و جهت‌های مختلف.....	۶۸۷
تطبیق چیدمان با استفاده از Visual State Manager.....	۶۹۵
تعریف یک چیدمان برای حالت نمایش Snapped.....	۶۹۶
استفاده از Visual State Manager برای مشخص کردن یک چیدمان.....	۷۰۱
اعمال شیوه‌ها روی اینترفیس کاربری.....	۷۰۴
تعریف شیوه برای فرم Customers.....	۷۰۴
خلاصه فصل ۲۵.....	۷۱۴
فصل ۲۶: نمایش و جستجو برای داده‌ها در یک اپلیکیشن ویندوز استور.....	۷۱۵
پیاده‌سازی الگوی Model-View-ViewModel.....	۷۱۶
نمایش داده‌ها با استفاده از اتصال داده‌ای.....	۷۱۷
استفاده از اتصال داده‌ای برای نمایش اطلاعات Customer.....	۷۱۷
تغییر داده‌ها با استفاده از اتصال داده‌ای.....	۷۲۳
پیاده‌سازی اتصال داده دو طرفه برای تغییر اطلاعات Customer.....	۷۲۳
پیاده‌سازی اینترفیس INotifyPropertyChanged در کلاس Customer.....	۷۲۵
استفاده از اتصال داده‌ای با یک کنترل ComboBox.....	۷۲۸
پیاده‌سازی اتصال داده‌ای برای کنترل‌های ComboBox.....	۷۲۹
ایجاد یک ViewModel.....	۷۳۱
ایجاد یک ViewModel برای مدیریت اطلاعات Customer.....	۷۳۱
اضافه کردن دستورات به یک ViewModel.....	۷۳۵
پیاده‌سازی کلاس Command.....	۷۳۶
اضافه کردن دستورات NextCustomer و PreviousCustomer به کلاس ViewModel.....	۷۴۰
اضافه کردن دکمه‌های Next و Previous به فرم Customers.....	۷۴۴
خلاصه فصل ۲۶.....	۷۴۸
فصل ۲۷: آشنایی با WINDOWS PRESENTATION FOUNDATION.....	۷۴۹
ایجاد یک برنامه WPF.....	۷۵۰
ساختن برنامه WPF.....	۷۵۱
ایجاد پروژه BellRingers.....	۷۵۱

۷۵۱	بررسی فرم و چیدمان Grid
۷۵۶	اضافه کردن یک تصویر پس‌زمینه به فرم
۷۵۸	ایجاد یک شیوه برای مدیریت ظاهر کنترل‌های روی فرم
۷۶۴	اضافه کردن کنترل‌ها به فرم
۷۶۴	استفاده از کنترل‌های WPF
۷۶۵	اضافه کردن کنترل‌ها به فرم
۷۷۰	اعمال شیوه روی کنترل‌ها و آزمایش فرم
۷۷۳	تغییر دینامیک مشخصات
۷۷۳	ایجاد متد Reset
۷۷۸	اداره رویدادها در یک فرم WPF
۷۷۸	پردازش رویدادها در فرم‌های ویندوز
۷۷۸	اداره رویدادهای Click برای دکمه‌های Clear و Add
۷۸۱	اداره کردن رویداد Closing برای فرم
۷۸۳	خلاصه فصل ۲۷

PDF Compressor Free Version

بخش ۱

آشنایی با ویژوال C# و
مایکروسافت ویژوال
استودیو ۲۰۱۳

PDF Compressor Free Version

PDF Compressor Free Version

فصل ۱

به #C خوش آمدید!

#C Development

Microsoft
Visual Studio



این فصل مقدمه ای برای ویژوال استودیوی ۲۰۱۳ است، یعنی محیط برنامه‌نویسی و ابزارهایی که برای کمک به ایجاد برنامه‌های ویندوز طراحی شده‌اند. ویژوال استودیوی ۲۰۱۲ ابزاری ایده‌آل برای نوشتن کدهای C# است و امکانات زیادی فراهم آورده که در این کتاب به تدریج با آن‌ها آشنا می‌شوید. در این فصل، از ویژوال استودیوی ۲۰۱۳ برای ایجاد تعدادی برنامه C# ساده استفاده کرده و می‌بینید که چگونه پروژه‌هایی برای ویندوز ایجاد کنید.

شروع به برنامه‌نویسی با محیط ویژوال استودیوی ۲۰۱۳

ویژوال استودیوی ۲۰۱۳ یک محیط برنامه‌نویسی با امکانات بسیار زیاد بوده که برای ایجاد پروژه‌های بزرگ و کوچک C# به آن‌ها نیاز خواهید داشت. حتی می‌توانید پروژه‌هایی ایجاد کنید که حاوی ماژول‌های نوشته شده در زبان‌های برنامه‌نویسی متفاوت مانند C++، Visual Basic و F# هستند. در اولین تمرین، محیط برنامه‌نویسی ویژوال استودیو را باز کرده و یاد می‌گیرید که چگونه یک برنامه برای کنسول ایجاد کنید.

نکته

یک برنامه کنسول (Console Application)، برنامه‌ای است که به جای داشتن یک رابط کاربر گرافیکی (Graphical User Interface یا GUI تحت خط فرمان اجرا می‌شود.

ایجاد یک برنامه کنسول در ویژوال استودیوی ۲۰۱۳

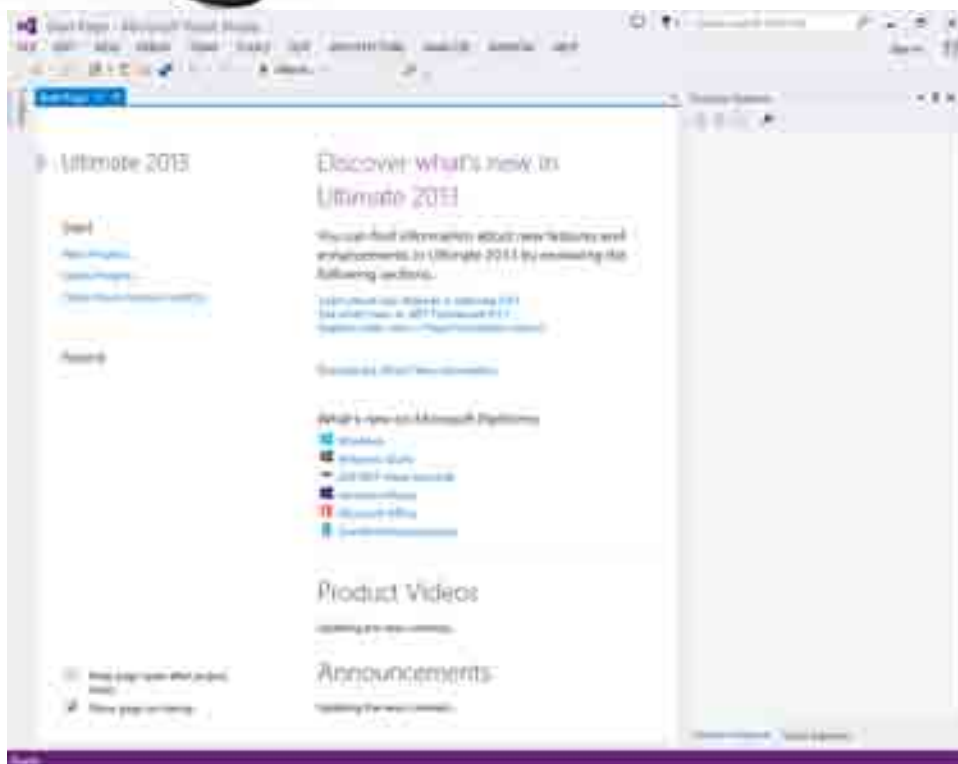
اگر از ویندوز ۸ استفاده می‌کنید، در صفحه استارت روی کاشی Visual Studio 2013 کلیک کنید. ویژوال استودی ۲۰۱۳ اجرا شده و صفحه Start خود را نمایش می‌دهد که در شکل ۱-۱ نشان داده شده است (صفحه Start شما ممکن است متفاوت باشد که به نسخه ویژوال استودیوی شما بستگی دارد).

نکته

اگر این اولین باری است که ویژوال استودیو ۲۰۱۳ را اجرا می‌کنید، ممکن است پنجره‌ای ظاهر شده که از شما می‌خواهد تا تنظیمات اولیه محیط توسعه برنامه را انتخاب کنید. ویژوال استودیوی ۲۰۱۳ خود را با توجه به زبان برنامه‌نویسی انتخاب شده توسط شما تغییر می‌دهد. پنجره‌ها و انتخاب‌های درون محیط توسعه یکپارچه (IDE: Integrated Development Environment) ویژوال استودیو از مقادیر پیش‌فرض برای زبان برنامه‌نویسی مورد نظر شما استفاده می‌کنند. در لیست Visual C# Development Settings را انتخاب کرده و سپس روی دکمه Start Visual Studio را انتخاب کنید. بعد از یک تاخیر کوتاه، محیط IDE ویژوال استودیو ظاهر می‌شود.

نکته

اگر از ویژوال استودیوی ۲۰۱۳ در ویندوز ۸٫۱ استفاده می‌کنید، هنگام باز کردن پروژه‌های مخصوص ویندوز ۸ این کتاب، ممکن است پیغامی مبنی بر نبودن پشتیبانی از بعضی امکانات قدیمی‌تر ویندوز ۸ ظاهر شود. در این حالت، روی گزینه‌ای کلیک کنید که پشتیبانی برای اپلیکیشن‌های ویندوز ۸ را نصب می‌کند. برای نصب، به دیسک ویژوال استودیو نیاز دارید.



شکل ۱-۱: صفحه Start ویژوال استودیوی ۲۰۱۳.

اما اگر از ویندوز ۷ استفاده می‌کنید، برای اجرای ویژوال استودیوی ۲۰۱۳ مراحل زیر را دنبال کنید:

۱. در نوار وظیفه ویندوز روی دکمه استارت کلیک کرده و سپس روی All Programs کلیک کرده و سپس روی گروه برنامه Microsoft Visual Studio 2013 کلیک کنید.
 ۲. در گروه باز شده روی Visual Studio 2013 کلیک کنید.
- ویژوال استودیو باز شده و صفحه Start را نمایش می‌دهد.

نکته

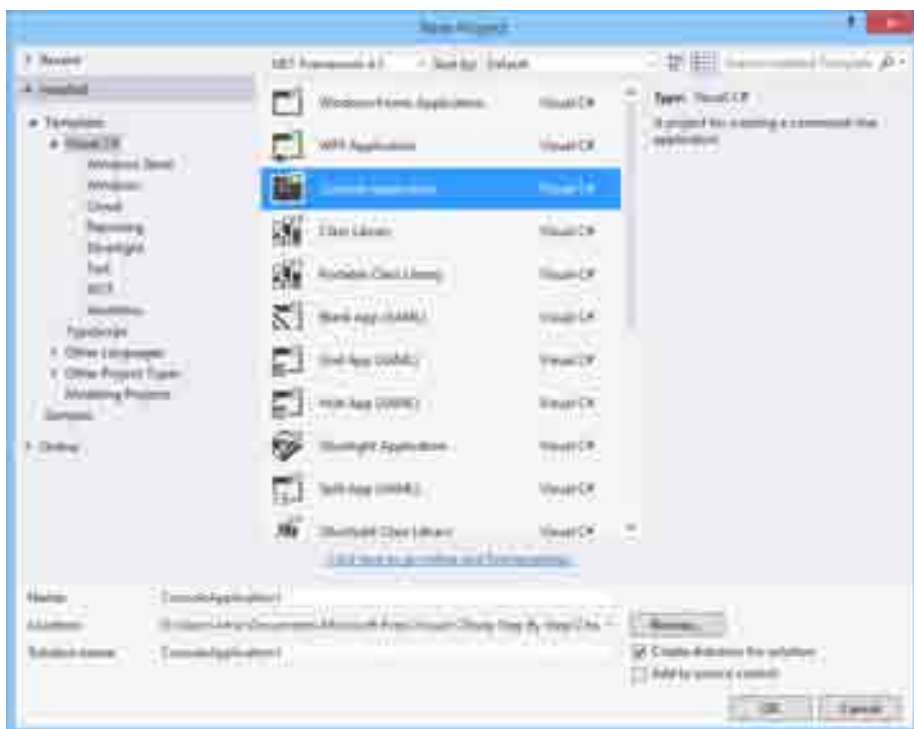
برای جلوگیری از تکرار در این کتاب، منظور من از ویژوال استودیو، نسخه ۲۰۱۳ آن است.

اکنون کارهای زیر را برای ایجاد یک برنامه کنسول جدید انجام دهید:

۱. در منوی File روی New و سپس روی Project کلیک کنید.

با این کار پنجره New Project باز می‌شود. این پنجره الگوهایی را لیست کرده است که می‌توانید از آن‌ها به عنوان نقطه شروع ایجاد یک برنامه استفاده کنید. این پنجره الگوها را با توجه به زبان برنامه‌نویسی که استفاده می‌کنید و نوع برنامه مورد نظر شما دسته‌بندی کرده است.

۲. در پنل سمت چپ و در پایین Templates (الگوها) روی Visual C# کلیک کنید. در پنل وسط مطمئن شوید که لیست بالای پنل عبارت 4.5.NET Framework را نشان می‌دهد. سپس روی آیکن Console Application کلیک کنید (شکل ۱-۲).



شکل ۱-۲: انتخاب نوع برنامه.

۳. در کادر Location، مسیر زیر را تایپ کنید. YourName را با نام کاربری ویندوز خود جایگزین کنید.

C:\Users\YourName\Documents\Microsoft Press\Visual CSharp Step By Step\Chapter 1.

نکته

برای صرفه‌جویی در تایپ، از این به بعد به مسیر C:\Users\YourName\Documents پوشه Documents یاد می‌کنم.

نکته

اگر پوشه‌ای که مشخص می‌کنید وجود نداشته باشد، ویژوال استودیو آن را برای شما ایجاد می‌کند.

۴. در کادر Name عبارت TestHello را جایگزین نام ConsoleApplication1 کنید.

۵. مطمئن شوید که گزینه Create Directory for Solution (ایجاد پوشه برای حل) انتخاب شده و سپس روی OK کلیک کنید.

ویژوال استودیو پروژه را با استفاده از الگوی Console Application ایجاد کرده و کد اولیه برای پروژه را نمایش می‌دهد (شکل ۱-۳).



شکل ۱-۳: کد اولیه ایجاد شده توسط ویژوال استودیو.

نوار منو در بالای صفحه اجازه دسترسی به ابزارهایی را می‌دهد که از آن‌ها در محیط برنامه‌نویسی استفاده خواهید کرد. برای دسترسی به منوها و دستورات درون آن‌ها همانند دیگر برنامه‌های ویندوز از صفحه کلید و ماوس استفاده می‌کنید. نوار ابزار نیز در پایین نوار منو قرار داشته و دکمه‌های میانبری برای اجرای دستورات پرکاربردتر را نشان می‌دهد.

پنجره Code and Text Editor قسمت اصلی صفحه را اشغال کرده و محتویات فایل‌های منبع را نشان می‌دهد. در یک پروژه چند فایل، وقتی بیش از یک فایل را ویرایش می‌کنید، هر فایل منبع دارای سربرگ مخصوص به خود با نام فایل منبع است. می‌توانید روی سربرگ آن فایل کلیک کرده تا فایل منبع آن ظاهر شده و بتوانید آن را ویرایش کنید.

پنل Solution Explorer در سمت راست پنجره قرار دارد که در شکل ۱-۴ نشان داده شده است.



شکل ۱-۴: پنل Solution Explorer.

پنل Solution Explorer نام فایل‌های مربوط به پروژه را به همراه موارد دیگر نشان می‌دهد. می‌توانید روی یک فایل در این پنل دوبار کلیک کرده تا محتویات آن در پنجره Code and Text Editor ظاهر شود.

قبل از این که کدی را بنویسید، ابتدا فایل‌های لیست شده در پنل Solution Explorer را بررسی کنید که ویژوال استودیو آن‌ها را به عنوان قسمتی از پروژه شما لیست کرده است:

◀ Solution “Test Hello”: این بالاترین فایل حل (Solution File) است. هر برنامه دارای یک فایل حل است. یک حل می‌تواند شامل یک یا تعداد بیشتری پروژه باشد و ویژوال استودیو فایل حل را ایجاد کرده تا به سازماندهی پروژه شما کمک کند. اگر از ویندوز اکسپلورر استفاده کرده و نگاهی به درون پوشه Visual C# Step By Step\Chapter 1\ TestHello ببینید، می‌بینید که نام واقعی این فایل TestHello.sln است.

◀ TestHello: این فایل پروژه C# است. هر فایل پروژه به یک یا تعداد بیشتری فایل حاوی کد منبع و تعدادی اجزا دیگر مانند تصاویر گرافیکی برای پروژه ارجاع می‌دهد. همه کدهای منبع شما در یک پروژه باید با یک زبان برنامه‌نویسی یکسان نوشته شوند. در ویندوز اکسپلورر این فایل با نام TestHello.csproj در پوشه TestHello پروژه است.

◀ Properties: این یک پوشه در پروژه TestHello است. اگر آن را باز کنید (روی فلش کنار متن آن کلیک کنید)، می‌بینید که حاوی یک فایل با نام AssemblyInfo.cs است. این یک فایل خاص است که می‌توانید از آن برای اضافه کردن خاصیت‌هایی به یک برنامه استفاده کنید مانند نام نویسنده، تاریخ نوشته شدن برنامه و مانند آن. می‌توانید خاصیت‌های بیشتری را مشخص کرده تا چگونگی اجرای برنامه را نیز تغییر دهید. چگونگی انجام این کار فراتر از سطح این کتاب است.

◀ References: این پوشه حاوی ارجاع‌هایی به کتابخانه‌هایی از کدهای کامپایل شده است که برنامه شما می‌تواند از آن‌ها استفاده کند. وقتی کد C# شما کامپایل می‌شود، به یک کتابخانه تبدیل شده و یک نام منحصر به فرد دریافت می‌کند. در فریم‌ورک دات‌نت، این کتابخانه‌ها با نام اسمبلی (Assembly) شناخته می‌شوند. برنامه‌نویس‌ها از اسمبلی‌ها استفاده کرده تا عملکردهای مفیدی که نوشته‌اند را بسته‌بندی کنند و بتوانند آن‌ها را به دیگر برنامه‌نویس‌ها بدهند تا آن‌ها نیز بتوانند از آن امکانات استفاده کنند. اگر پوشه References را باز کنید، یک سری ارجاع پیش‌فرض می‌بینید که ویژوال استودیو به پروژه شما اضافه کرده است این اسمبلی‌ها امکان دسترسی به امکانات پرکاربرد فریم‌ورک دات‌نت را فراهم آورده و مایکروسافت آن‌ها را همراه ویژوال استودیو ارائه کرده است.

◀ App.config: این فایل پیکربندی برنامه است. این فایل دلخواه است و لازم نیست همیشه وجود داشته باشد. می‌توانید تنظیماتی را در آن مشخص کنید که می‌خواهید برنامه شما بتواند در زمان اجرا از آن‌ها استفاده کند تا بتواند رفتار خود را تعیین کند (به عنوان مثال فریم‌ورک دات‌نتی که باید برای اجرای برنامه استفاده شود). در فصل‌های بعدی این کتاب جزئیات بیشتری درباره این فایل را خواهید دید.

◀ Program.cs: این یک فایل منبع C# بوده و هنگام ایجاد پروژه در پنجره Code and Text Editor نمایش داده می‌شود. کد برنامه کنسول خود را در این فایل می‌نویسید. این فایل شامل مقداری کد است که ویژوال استودیو به صورت خودکار برای شما قرار داده است که اندکی بعد آن‌ها را بررسی خواهیم کرد.

نوشتن اولین برنامه

فایل Program.cs یک کلاس با نام Program را تعریف کرده که حاوی یک متد (Method) با نام Main است. در C# همه کد اجرایی باید درون یک متد باشند و همه متدها نیز باید به یک کلاس یا استراکت (Struct) تعلق داشته باشند. درباره کلاس‌ها در فصل ۷ و درباره استراکت‌ها در فصل ۹ یاد می‌گیرید.

متد Main به عنوان نقطه شروع برنامه تعیین شده است. این متد باید به گونه مشخص شده درون کلاس Program تعریف شده باشد و گزینه فریم‌ورک دات‌نت ممکن است هنگامی که کد شما را اجرا می‌کند آن را به عنوان نقطه شروع تشخیص ندهد. (متدها را با جزئیات در فصل ۳ و متدهای استاتیک را در فصل ۷ خواهید دید.)

نکته مهم

C# یک زبان حساس به کوچکی و بزرگی حروف است. بنابراین حتماً باید Main را با حرف M بزرگ تایپ کنید.

در تمرین‌های بعدی، کدی را می‌نویسید که در آن پیغام Hello World! در پنجره کنسول نمایش داده می‌شود؛ برنامه کنسول Hello World را ایجاد و آن را اجرا می‌کنید؛ و یاد می‌گیرید که چگونه فضاهای نام برای تقسیم عناصر کد استفاده می‌شوند.

نوشتن کد با استفاده از ابزار IntelliSense مایکروسافت

۱. در پنجره Code and Text Editor فایل Program.cs را باز کرده، نشانگر را در متد Main بعد از براکت باز { قرار داده و سپس کلید Enter را بزنید تا یک خط جدید ایجاد کنید.

۲. در خط جدید، کلمه Console را تایپ کنید؛ این نام یک کلاس دیگر است که توسط اسمبلی‌های ارجاع داده شده توسط برنامه شما فراهم آمده است. این کلاس متدهایی برای نمایش پیغام‌ها در پنجره کنسول و خواندن ورودی از صفحه کلید دارد.

وقتی حرف C شروع کلمه Console را تایپ می‌کنید، یک لیست IntelliSense ظاهر می‌شود که در شکل ۱-۵ نشان داده شده است.

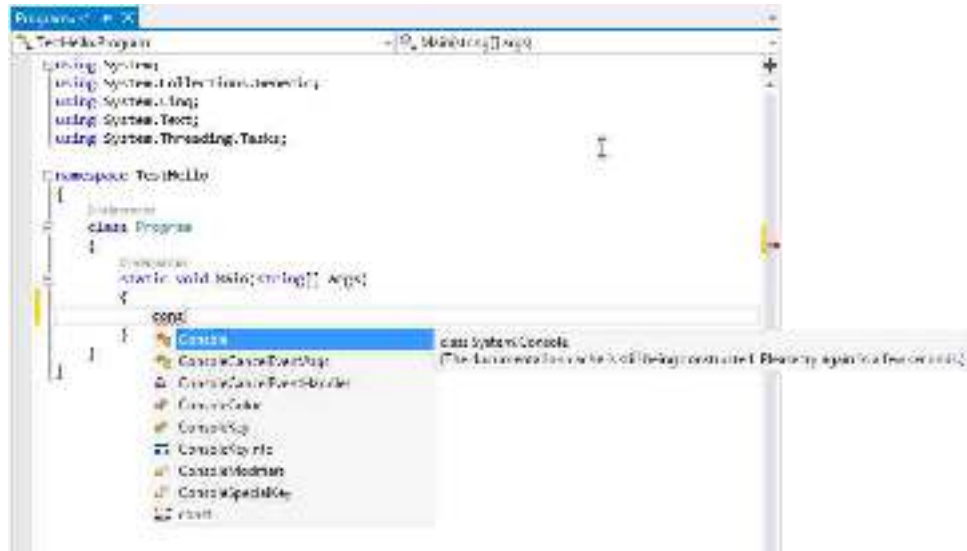
این لیست حاوی همه کلمه‌های کلیدی C# و انواع داده‌ای است که برای این محتوا معتبر هستند. می‌توانید به تایپ ادامه داده یا این که در لیست اسکرول کرده و دوبار روی Console درون لیست کلیک کنید. یا بعد از این که Cons را تایپ کردید، IntelliSense به صورت خودکار مورد Console را های لایت می‌کند. اکنون می‌توانید کلید Tab یا Enter را بزنید تا آن را انتخاب کنید.

اکنون متد Main باید به این صورت باشد:

```
static void Main(string[] args)
{
    Console
}
```

نکته

Console یک کلاس داخلی در C# است.



شکل ۱-۵: ابزار IntelliSense.

۳. یک نقطه بعد از Console تایپ کنید. یک لیست IntelliSense ظاهر شده که متدها، مشخصات و فیلدهای کلاس Console را نمایش می‌دهد.

۴. در لیست به پایین اسکرول کرده، سپس WriteLine را انتخاب کنید و Enter را بزنید. یا می‌توانید به تایپ کاراکترهای W، L، e، t، i، r، و ادامه داده تا WriteLine انتخاب شده و سپس Enter را بزنید.

لیست بسته شده و کلمه WriteLine به فایل منبع اضافه می‌شود. اکنون متد Main باید به این صورت باشد:

```
static void Main(string[] args)
{
    Console.WriteLine
}
```

۵. یک پرانتز باز (را تایپ کنید. یک کادر کوچک IntelliSense ظاهر می‌شود.

این کادر پارامترهایی که متد WriteLine می‌تواند دریافت کند را نشان می‌دهد. در حقیقت WriteLine یک متد اورلود شده (Overloaded) است، یعنی کلاس Console حاوی بیش از یک متد با نام WriteLine است- در حقیقت دارای ۱۹ نسخه متفاوت از این متد است. هر نسخه متد WriteLine را می‌توان برای نمایش خروجی نوع متفاوتی از داده‌ها به کار برد. (فصل ۳ متدهای اورلود شده را با جزئیات بیشتری توضیح می‌دهد.) اکنون Main باید به این شکل باشد:

```
static void Main(string[] args)
{
    Console.WriteLine(
}
```